

# Re: Algorithm suggestions

---

*Source:* <http://www.derkeiler.com/Newsgroups/sci.crypt/2006-09/msg00959.html>

---

- *From:* "John E. Hadstate" <jh113355@xxxxxxxxxxxx>
  - *Date:* Tue, 26 Sep 2006 19:31:18 -0400
- 

"Geoffrey Summerhayes" <sumrnot@xxxxxxxxxxxx> wrote in message  
[news:1159294685.907516.118210@xx](mailto:news:1159294685.907516.118210@xx)

I'm looking for a relatively simple algorithm to add a second layer of protection to a transmitted packet.

The data to be encrypted is less than 256 bytes per packet, with a fair amount of similarity in the data, multiple senders, each can have their own key, the final packet is sent using SSL.

The main reason for the additional layer is to deter packet sniffing between the sender and the SSL device.

The units are fairly limited, it would be nice if the same function worked for the encode and decode. Any suggestions?

Yes. You need to think about what you really want from this security and what you are willing to give up to get it. For example:

1. Does your attack model envision anything except snooping? For example, will your attacker try to become an active man-in-the-middle, seeking to replace all or part of one device's data with data of his own choosing? Will your attacker seek to corrupt portions of your data packet (without necessarily knowing what the original content was or what the decrypted content will be)?

2. How valuable is the data that the attacker could obtain by snooping? For how long will it maintain its value? How

## Re: Algorithm suggestions

much could it cost if an attacker could substitute meaningful data into the data packet undetected?

3. How limited is "fairly limited"? Do the devices support TCP/IP stacks? Do they run embedded Windows or Linux? How much CPU power do they have and how much of that is available? What cryptographic support is available from the O/S or your programming language? Will your app be implemented in Java using a current Sun JDK?

Rolling your own "simple algorithm to add a second layer of protection" will be, at best, a waste of time. At worst it will engender a false sense of security in those who are depending on true security and set the stage for a rude awakening in the future.

If you must implement at the nuts-and-bolts level in C, use a portable, well-tested library like libtomcrypt. The advantages of this are too numerous to list, the most important being that you can get support through this newsgroup if you need it. If you are working in Java, use the Sun JDK's Crypto API. Either way, you can at least be sure you're using a reputable cipher, even if you might not be using it in the most secure way possible.